LABO 1: Javascript syntax & OO concepts

Voorkennis

Deze sessie gaat over de taal Javascript zelf: de basissyntax, functies, objecten, constructoren, namespacing, scope… We wachten nog even met de toepassing op web pagina's, kwestie van een duidelijk onderscheid te maken tussen taal en toepassing. Zorg ervoor dat je de theorieles vooraf over Javascript Syntax goed bestudeerd hebt, en minstens vertrouwd bent met volgende Javscript concepten:

- Creatie variabelen, incl. functies en objecten
- Onderscheid tussen literals en constructors
- Typeof en instanceof
- Type conversies (impliciet en manueel)
- Anonieme functies en objecten
- Functies als argument, arguments array, object parameter
- Javascript constructors, prototype, namespacing
- Scope en closures

De allerbelangrijkste slides voor dit labo komen uit de theorieles Javascript Syntax

> Running Javascript
>> Read from the shell
> Variables and constructors
>> Literals and constructors
>> NaN, undefined
> Types
>> Weak and dynamic typing
>> Type conversion
> Functions
>> Single object parameter
>> Default parameters
> Objects
>> OO in Javascript
>> Prototype property
>> Extending built-in classes
>> Namespaces
> Quick syntax recap
>> For iterations (1)
> Coding rules (volledig)

Doelstellingen

- Vertrouwd raken met de eigenheden van de taal Javascript

Werkwijze, timing en upload

Dit labo loopt over één sessie en moet op het einde van de labosessie geupload worden via Toledo. Upload één enkele zip met daarin je oplossingen. De docent kan vragen de opdrachten thuis verder af te werken – je kunt dan wat extra punten verdienen. Neem die dan gewoon het volgende labo mee; je hoeft niet opnieuw te uploaden.

Het huishoudelijk reglement is van toepassing.

Maak alle oefeningen in een console-omgeving, bijvoorbeeld Windows Scripting Host in Windows of JavaScriptCore in Mac.

Opgaves

1. Opwarmers

a. Maak een klassiek 'Hello World' script (oefening op werken met scripting host)

b. Maak een scriptje dat twee getallen inleest en het gemiddelde teruggeeft (oefening op inlezen)

c. Vorm het scriptje nu zo om dat de getallen via CL arguments ingelezen worden; er moet een willekeurig aantal getallen opgegeven kunnen worden (oefening op argumenten)

d. Schrijf een functie `countInt(arr)` die het aantal gehele getallen in een array teruggeeft. Definieer een array van een 6-tal waarden (mix van booleans, getallen en strings), en test je functie (oefening op functies, arrays en prototype)

   Tip: bij een geheel getal is de waarde gelijk aan de naar `int` geparste versie

2. Objecten en arrays

Gegeven volgende personen:

Astrid:
    Naam: Astrid Galli
    Balans: 2500
    Vrienden: Bilal Azzouti, Crista Bracke, Duncan Reck
Bilal:
    Naam: Bilal Azzouti

Balans: 3000
Vrienden: An Cornelis, Duncan Reck

Claude:

Naam: Claude Chen
Balans: -300
Vrienden: Erwin Smith, Astrid Galli, Francis Alys

a. Maak drie objecten aan voor deze personen (als object literal, dus zonder constructor).

b. Schrijf een methode `createCouple` die twee personen combineert tot een nieuw object met de properties `name`, `balance` en `friends`, en dit object teruggeeft via `return`. De waarden ervan zijn de combinaties van de twee. Let bij de vrienden op dat je geen dubbels hebt, en dat de namen van de twee personen zelf niet meer in de gemeenschappelijke vriendenlijst voorkomen. Voor Astrid en Bilal zou het resultaat b.v. het volgende zijn:

```
{
    name: 'Astrid Galli & Bilal Azzouti',
    balance: 5500,
    friends: ['An Cornelis', 'Crista Bracke', 'Duncan Reck']
}
```

c. Voorzie standaardwaarden voor de balans en vrienden properties van de twee personen (neem 0 resp. een lege array). Anders gezegd: als je bij één van de twee personen de balans of vrienden verwijdert, mag dit geen fout genereren.

(oefening op creatie en gebruik objecten)

3. Constructors

Maak een constructor `Disc` aan met een volgende properties:

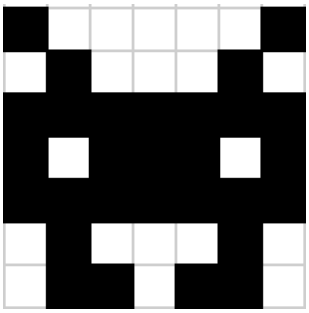
```
radius
PI
getCircumference()
getArea()
```

Maak als test een schijf aan en geef omtrek en oppervlakte weer.

4. Grote oefening

De theorie

In game programming is een *sprite* een grafische voorstelling van een karakter of voorwerp in het spel. In zijn oervorm bestaat het uit een raster van puntjes:



Je zou in principe elke sprite kunnen opslaan in een matrix:

```
var sprite1 = [
  [1, 0, 0, 0, 0, 0, 1],
  [0, 1, 0, 0, 0, 1, 0],
  [1, 1, 1, 1, 1, 1, 1],
  [1, 0, 1, 1, 1, 0, 1],
  [1, 1, 1, 1, 1, 1, 1],
  [0, 1, 0, 0, 0, 1, 0],
  [0, 1, 1, 0, 1, 1, 0]
];
```

Dit zijn 155 karakters per sprite. Vandaag is dat niet zo'n probleem, maar in de eerste Arcade omgevingen, waar vaak niet meer dan enkele kB geheugen beschikbaar waren, moesten kortere manieren gezocht worden. Dit kan door op te merken dat elke rij eigenlijk de binaire codering van een ASCII karakter voorstelt. De eerste rij bijvoorbeeld kan korter voorgesteld worden door karakter 'A':

$1000001_{bin} = 65_{dec}$ = ASCII karakter 'A'

Elke sprite kan dus voorgesteld worden door 7 karakters. Om niet-afdrukbare karakters te vermijden wordt ook nog een XOR 69 gebruikt, maar de theorie daarvan zou ons te ver leiden. Onthoud vooral dat elke sprite kan voorgesteld worden door 7 karakters. Bij de oefening is een Excel meegegeven die de omzetting doet.

Om een sprite te tekenen moet de string van 7 karakters weer gedecodeerd worden naar een 2D matrix van 1 en 0. Een codefragment:

```
// encoded sprite
var encoded = 'xxxxxxx'; // copy-paste hier de string uit de Excel

// 7x7 grid of 1 and 0
var grid = [];

// loop each character
for (var i = 0; i < encoded.length; i++) {
   // get ascii code
   var code = encoded[i].charCodeAt(0);

   // XOR 69
   code = code ^ 69;

   // pad zeros
   var row = ('00000000' + code.toString(2));
   row = row.substr(row.length - 7).split('');
```

```
    // push row to grid
    grid.push(row);
}
```

De opgave

Gegeven volgend geraamte van een Sprite constructor:

```
// define sprite constructor
var Sprite = function(encoded) {
    // encoded
    this.encoded = encoded;

    // grid
    this.grid = [];

    // init function: convert encoded to 2D grid of 1 and 0
    this.init = function() {
        // ...
    }

    // toString function: return string representation
    // using char0 for 0 and char1 for 1, e.g. '░' and '█'
    this.toString = function(char0, char1) {
        // ...
    }

    // start init
    this.init();
}
```

a. Vul de constructor aan; test het door een sprite aan te maken (voor de encoded versie zie bijgeleverde 'sprite encoder.xslx'; eventueel kun je zelf een sprite tekenen) en het af te drukken via `toString()`.

b. (optioneel, enkel voor de echte freaks) Maak nu een constructor `Gameboard` met standaard 50 kolommen en 20 rijen waaraan je objecten van type `Sprite` kunt toevoegen. Voorzie methodes `init()`, `addSprite()`, `clear()` en `toString()`